Sistemas de Archivos Distribuidos

Prof. Mariela Curiel

Bibliografía:
Sistemas Distribuidos (Tanembaum),

Introducción

- Un sistema de archivos distribuidos permite a los procesos el acceso transparente y eficiente de archivos que permanecen en servidores remotos.
- Se encargan de la organización, almacenamiento, recuperación, nominación, compartimiento y protección de los archivos.
- Proporcionan una interfaz de programación que oculta a los programadores los detalles de localización y asignación del almacenamiento

Ventajas

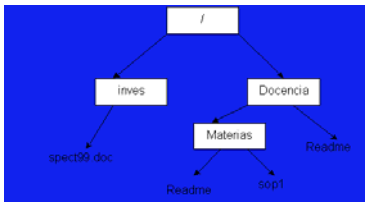
- Un usuario puede acceder a sus mismos archivos desde diferentes máquinas
- Usuarios diferentes desde diferentes máquinas pueden acceder a los mismos archivos (compartir archivos)
- Más fácil de administrar: sólo un servidor o grupo de servidores.

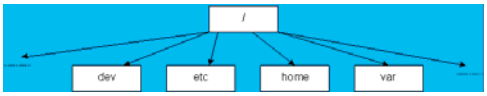
Definiciones

- Archivo

En la mayoría de los sistemas de operación, los datos y programas ejecutables están organizados en archivos para garantizar un almacenamiento permanente.

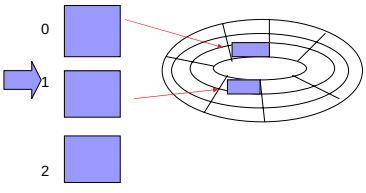
En UNIX un archivo es una secuencia de bytes. Están jerarquicamente organizados en un grafo de nombres donde los nodos representan directorios y archivos.





knlknlknk}+
ñlñl{k{ñññ

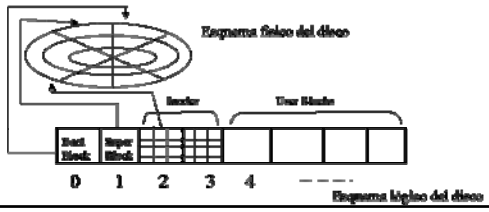
llñ,-.,-,,,,
ññ,Ñ,,,,Ñ,ñ,ñ,ñ,

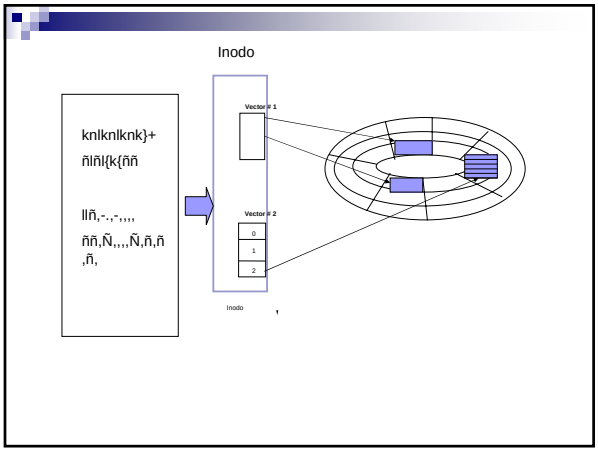


Conceptualmente un archivo UNIX es una secuencia lineal de bytes. El kernel no soporta estructuras de bajo nivel, el registro en un byte. Este archivo de distribuye en varios bloques dentro del disco.

Definiciones

- Sistema de Archivos: es el conjunto de archivos, atributos, directorios y bloques de datos que se implementan juntos como un dispositivo de bloques lógico.





Definiciones

Servidor de archivos:

- Proceso que se ejecuta en alguna máquina y ayuda a implantar el manejo de archivos.
- Se ejecuta en el espacio de usuario, por lo que el sistema puede contener varios Servidores de Archivos con servicios de archivos diferentes.
- Puede haber 1 o más servidores de archivos, pero ésto debe ser transparente a los clientes.

Definiciones

Servicios de Archivos

- Especificación de los servicios o interfaz que el servidor de archivos (SA) ofrece a sus clientes

File System Model (1)

- Figure 11-3. An incomplete list of file system operations supported by NFS.

Operation	v3	v4	Description
Create	Yes	No	Create a regular file
Create	No	Yes	Create a nonregular file
Link	Yes	Yes	Create a hard link to a file
Symlink	Yes	No	Create a symbolic link to a file
Mkdir	Yes	No	Create a subdirectory in a given directory
Mknod	Yes	No	Create a special file
Rename	Yes	Yes	Change the name of a file
Remove	Yes	Yes	Remove a file from a file system
Rmdir	Yes	No	Remove an empty subdirectory from a directory

File System Model (2)

- Figure 11-3. An incomplete list of file system operations supported by NFS.

Operation	v3	v4	Description
Open	No	Yes	Open a file
Close	No	Yes	Close a file
Lookup	Yes	Yes	Look up a file by means of a file name
Readdir	Yes	Yes	Read the entries in a directory
Readlink	Yes	Yes	Read the path name stored in a symbolic link
Getattr	Yes	Yes	Get the attribute values for a file
Setattr	Yes	Yes	Set one or more attribute values for a file
Read	Yes	Yes	Read the data contained in a file
Write	Yes	Yes	Write data to a file

Desafíos de un Servidor de Archivos

Transparencia:

- **De acceso**: no hay preocupación de la distribución de los archivos. Los programas deben acceder de igual forma archivos locales y remotos.
- **De localización**: Los archivos deben poder cambiarse sin que cambie el nombre.

Desafíos

Escalabilidad: El servicio se debe poder extender incrementalmente para atender a un amplio rango de cargas.

Tolerancia a fallas: Clientes y servidores deben operar correctamente ante fallas.

Consistencia: ante el acceso concurrente de diferentes usuarios.

Seguridad: mecanismos de control de acceso y autenticación

Eficiencia: el desempeño debe ser similar a sistemas de archivos locales.

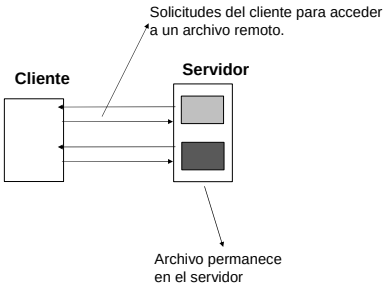
Desafíos

- Heterogeneidad de Hardware y de Sistemas de Operación.

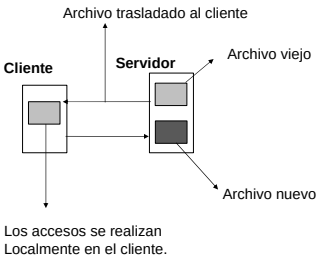
Arquitecturas: Cómo se organizan?

- Arquitecturas Cliente-Servidor
- Sistemas de Archivo Distribuidos basados en Cluster

- Modelo de Acceso Remoto



- Modelo Carga y Descarga.



Arquitectura Cliente-Servidor

- Uno de los sistemas de archivos más populares que trabaja con la arquitectura de acceso remoto es NFS (Network File System).
- La idea básica de NFS es que cada servidor de archivos proporcione una visión estandarizada (interfaz) de su sistema de archivos local, independientemente de la implementación de este último.

Arquitectura C/S

- El NFS cuenta con un protocolo de comunicación que permite a los clientes acceder a los archivos guardados en el servidor. Luego, es posible que un conjunto heterogéneo de procesos (quizás ejecutándose en máquinas diferentes con SO diferentes) compartan archivos.

Arquitectura C/S

- El modelo que se utiliza NFS es del de **servicio de archivos remoto**. Este modelo ofrece a los clientes un acceso transparente a un sistema de archivos gestionado por un servidor remoto.
- Los clientes desconocen la ubicación de los archivos
- Disponen de una interfaz, similar a la interfaz del sistema de archivos local. El servidor remoto es el responsable de implementar estas operaciones.

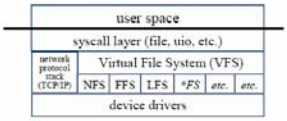
Implementación de NFS

- Un cliente accede al FS usando la interfaz local.
- La interfaz de Unix es reemplazada por VFS (Virtual File System), el cual actualmente es un estándar para comunicarse con Sistemas de Archivo diferentes. Todos los sistemas operativos modernos proporcionan un VFS.
- El VFS transfiere las operaciones a un sistema de archivos local o a un servidor remoto.

VFS: the Filesystem Switch

Sun Microsystems introduced the *virtual file system* interface in 1985 to accommodate diverse filesystem types cleanly.

VFS allows diverse *specific file systems* to coexist in a file tree, isolating all FS-dependencies in pluggable filesystem modules.



VFS was an internal kernel restructuring with no effect on the syscall interface.

Incorporates object-oriented concepts: a generic procedural interface with multiple implementations.

Based on abstract objects with dynamic method binding by type...in C.

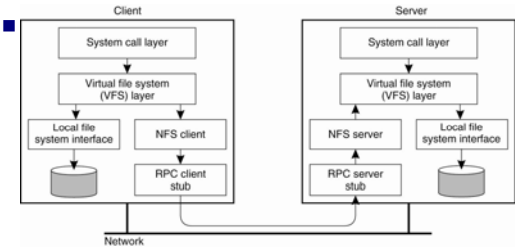
Other abstract interfaces in the kernel: device drivers, file objects, executable files, memory objects.

NFS Protocol

NFS is a network protocol layered above TCP/IP.

- Original implementations (and most today) use UDP datagram transport for low overhead.
  - Maximum IP datagram size was increased to match FS block size, to allow send/receive of entire file blocks.
  - Some newer implementations use TCP as a transport.
- The NFS protocol is a set of message formats and types.
  - Client issues a *request* message for a service operation.
  - Server performs requested operation and returns a *reply* message with status and (perhaps) requested data.

Client-Server Architectures (2)



Modelo de Sistema de Archivo NFS

- Es similar al ofrecido por los sistemas de archivos de UNIX
- Las interfaz es bastante similar.
- Para acceder a los archivos primero se busca su nombre en un servicio de asignación de nombres y éste provee el manejador de archivos asociado (file handle).

File System Model (1)

Operation	v3	v4	Description
Create	Yes	No	Create a regular file
Create	No	Yes	Create a nonregular file
Link	Yes	Yes	Create a hard link to a file
Symlink	Yes	No	Create a symbolic link to a file
Mkdir	Yes	No	Create a subdirectory in a given directory
Mknod	Yes	No	Create a special file
Rename	Yes	Yes	Change the name of a file
Remove	Yes	Yes	Remove a file from a file system
Rmdir	Yes	No	Remove an empty subdirectory from a directory

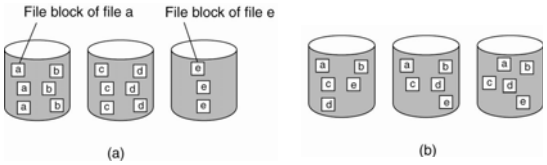
File System Model (2)

Operation	v3	v4	Description
Open	No	Yes	Open a file
Close	No	Yes	Close a file
Lookup	Yes	Yes	Look up a file by means of a file name
Readdir	Yes	Yes	Read the entries in a directory
Readlink	Yes	Yes	Read the path name stored in a symbolic link
Getattr	Yes	Yes	Get the attribute values for a file
Setattr	Yes	Yes	Set one or more attribute values for a file
Read	Yes	Yes	Read the data contained in a file
Write	Yes	Yes	Write data to a file

Sistemas de Archivos basados en Cluster

- *Distribución de un archivo entre varios servidores.*
- *Técnicas de Distribución de Archivos*: indican cómo se puede distribuir un archivo a través de múltiples servidores.
- Distribuyendo un archivo grande entre varios servidores es posible buscar sus diferentes partes en paralelo.
- Ejemplos: GFS, Lustre

Cluster-Based Distributed File Systems (1)



The difference between (a) distributing whole files across several servers and (b) striping files for parallel access.

Sistemas de Archivos basados en Cluster

- Esta organización (b) sólo funciona bien si el acceso a los datos es en paralelo. Se requiere que los datos tengan una estructura muy regular (e.j. una matriz densa, registros de un mismo tipo).
- Si se trata de archivos con tipos irregulares de datos o muchos tipos de estructuras de datos la distribución de archivos puede no ser una herramienta efectiva. En este caso lo mejor es guardar archivos completos en diferentes servidores.

Ejemplo: Google File System

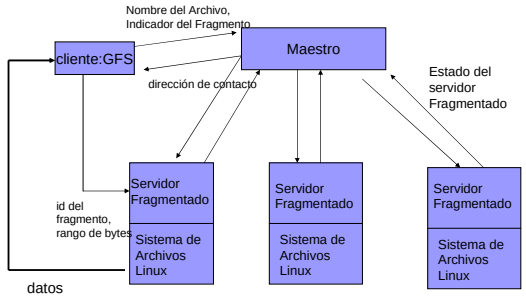
- Un sistema de archivos distribuido para grandes centros de datos.
- Compañías como Google y Amazon ofrecen servicios a clientes WEB cuyo resultado son lecturas y actualizaciones de un número masivo de archivos distribuidos a través de decenas de miles de computadoras. En estos entornos es de esperarse que en cualquier momento falle una computadora.

GFS

- Los archivos de Google tienden a ser muy grandes (varias Gbytes) y cada uno contiene objetos más pequeños.
- Las actualizaciones generalmente se realizan anexando datos en lugar de sobreescribiendo algunas partes del archivo.
- Las fallas del servidor son la regla en lugar de la excepción.

GFS

- Cada grupo GFS se compone de un servidor maestro con múltiples servidores fragmentados.
- Cada archivo se divide en fragmentos de 64Mbytes, tras de lo cual estos fragmentos se distribuyen en los servidores fragmentados.
- El maestro sólo se contacta para solicitar metadatos. La dirección de contacto que devuelve al cliente contiene toda la información necesaria para acceder al servidor fragmentado correcto y obtener el fragmento de archivo requerido.



GFS

- Los fragmentos se replican por tolerancia a fallos. Se replican de acuerdo con un esquema de respaldo primario.
- El maestro no intenta llevar una cuenta precisa de las ubicaciones de los fragmentos (original, réplicas), en su lugar se pone en contacto periódicamente con los servidores fragmentados para ver qué segmentos tienen guardados.

GFS

- La visión del maestro no tienen que estar consistente todo el tiempo.
- Los clientes GFS necesitan saber *lo que cree el servidor maestro acerca de donde están guardados los datos solicitados.* Como los fragmentos se replican, existe una alta probabilidad de que el fragmento esté disponible en alguno de los servidores fragmentados.

GFS: manejo de la escalabilidad

- La mayoría del trabajo lo hacen los servidores fragmentados. El Maestro no representa un cuello de botella.

Procesos

- Clientes
- Servidores: Con Estado- Sin Estado.
- Comunicación: Muchos de estos sistemas están basados en llamadas a procedimientos remotos (RPC). La razón principal es que el sistema sea independiente de los sistemas operativos subyacentes, redes y protocolos de transporte.

Procesos

Clientes:

- Se ejecutan en cada máquina cliente, a nivel de usuario.
- Mantienen información sobre las ubicaciones de los servidores de archivos.
- Manejan *caching* en el cliente

Procesos

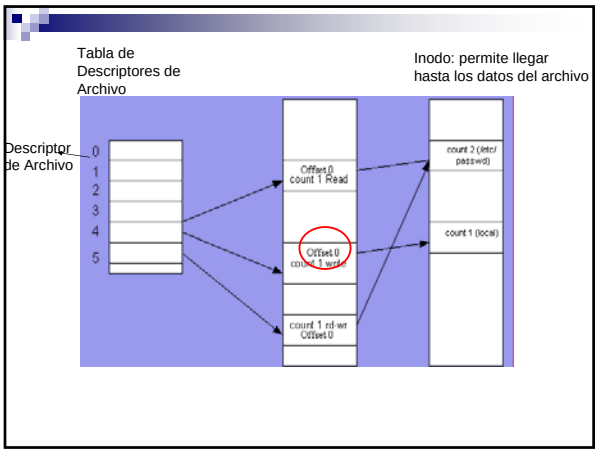
- **Servidores**: el aspecto más importante con respecto a los procesos servidores es si deben o no guardar el estado

Ejemplo


```
int s;

fd = open("/home/mcuriel/data.txt", R);
read(fd, &s, sizeof(int)); // lee 1
read(fd, &s, sizeof(int));// lee 2
```

-El servidor sabe qué file descriptor está asociado a qué archivo,
-El servidor también guarda la dirección u offset en el que se está leyendo.
-El servidor guarda información sobre el cliente....guarda estado.
-Las operaciones de lectura y escritura no son idempotentes, es decir, cada vez
-que se hacen se realizan sobre un registro distinto, depende del offset.



Ejemplo


```
int s;

open("/home/mcuriel/data.txt", R);
read("/home/mcuriel/data.txt, &s, sizeof(int), 1);
read("/home/mcuriel/data.txt, &s, sizeof(int), 2);
```

-En este caso el servidor no guarda información sobre qué proceso tiene abierto el archivo (puede ser que ni siquiera haya una operación de apertura).
-En cada operación el proceso cliente debe indicar sobre que archivo la va a hacer y el byte (o registro) a partir del cual comienza la lectura.
-En este caso las operaciones sí son idempotentes, siempre darán el mismo resultado no importa cuántas veces se ejecuten.

Procesos

- Servidores sin estado ("stateless")
  - Cuando un cliente envía una solicitud a un servidor, éste la lleva a cabo, envía la respuesta y elimina de sus tablas internas toda la información relativa a dicha solicitud. Por ejemplo, no registra si un archivo ha sido abierto previamente.
  - No guarda información del cliente entre solicitudes. No mantiene un registro de las operaciones que van dejando los clientes
  - Cada solicitud debe ser autocontenida. Operaciones idempotentes

Procesos

- Consecuencias:
  - Servidores y clientes son completamente independientes. (+)
  - Se reducen los estados inconsistentes debido a caídas de clientes y/o servidores (+)
  - Posible pérdida en las prestaciones, ejm.Un cliente no puede almacenar temporalmente un archivo para accederlo localmente, ya que el servidor tendría que almacenar esta información. (-)
  - La implementación de bloqueos es prácticamente imposible. En el caso de NFS V3 la realiza un gestor de bloqueos. (-)
  - "El estado" del servidor no crece con más clientes (+)

Procesos

Servidores con estado

• Los servidores guardan información del estado de los clientes entre solicitudes: ejm. tabla que asocia los descriptores de archivos con los archivos propiamente dichos.

•Pueden conocer qué datos están en el cache del cliente (permiten al cliente mantener copias locales de datos compartidos).

Proceso

- Consecuencias:
  - Mejor performance: mantener caches, lecturas adelantadas, mensajes mas cortos (+). La necesidad de cache en los clientes, para mejorar el desempeño en redes de área amplia fue una de las principales razones por las que NFS pasa de ser "sin estado" a "con estado".
  - El número de archivos abiertos tiene un límite (-)
  - Se puede manejar de forma más eficiente el bloqueo de archivos. (+)
  - Ante una caída del servidor hay que restaurar el estado.

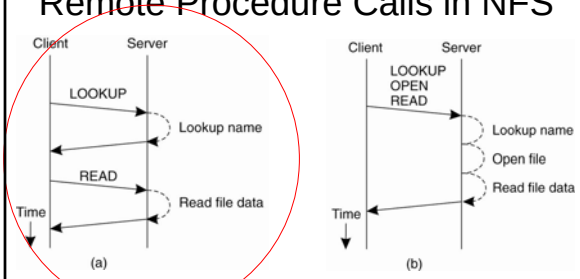
Comunicación

- No existe nada especial acerca de la comunicación entre procesos en sistemas de archivos distribuidos. La mayoría de estos sistemas están basados en llamadas a procedimientos remotos (RPC).
- Cada operación de NFS puede implementarse como una llamada a procedimiento remoto.

Comunicación

- Hasta NFSV3, el cliente era responsable de hacer la vida del servidor lo más fácil que le fuera posible manteniendo las solicitudes relativamente simples. Se hacía una operación a la vez.

Remote Procedure Calls in NFS

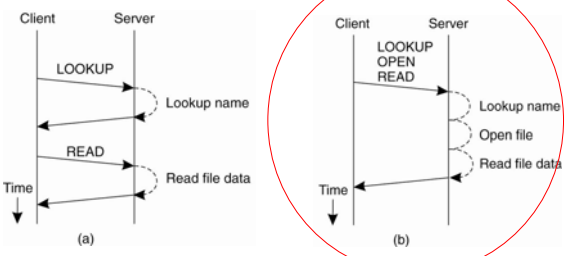


- Figure 11-7. (a) Reading data from a file in NFS version 3. (b) Reading data using a compound procedure in version 4.

Comunicación

- Este método requiere dos RPCs seguidas.
- Cuando se considera el NFS en un sistema de área ámplia el uso de muchos RPCs puede degradar el desempeño.
- NFSv4 soporta procedimientos compuestos, en los cuales varios RPCs pueden agruparse en una sola solicitud.

Remote Procedure Calls in NFS



- Figure 11-7. (a) Reading data from a file in NFS version 3. (b) Reading data using a compound procedure in version 4.

Comunicación

- Las operaciones compuestas se manejan en el orden en el que fueron solicitadas.
- Si existen operaciones concurrentes con otros clientes no se toman medidas para evitar conflictos.
- Si falla una de las operaciones ya no se ejecutan más operaciones y se regresan al cliente los resultados hasta ese momento.
- La interfaz debe ser especial??? (tarea).

Subsistema RPC2

- Se desarrolló como parte del sistema de archivos Coda (kistler y Satyanarayanan 1992).
- Ofrece RPC confiables encima de UDP
- Para enviar un RPC se crea un hilo que envía la solicitud de invocación al servidor y posteriormente se bloquea hasta que recibe una respuesta. Si la respuesta se está tardando, el servidor regularmente envía mensajes al cliente para informarle que sigue trabajando en la solicitud.

Subsistema RPC2

- Si el servidor deja de funcionar, tarde o temparno el hilo lo advierte y reportará la falla a la aplicación.
- RPC2 da soporte a efectos colaterales. Un efecto colateral es un mecanismo mediante el cual un cliente y un servidor pueden comunicarse usando un protocolo específico de una aplicación.

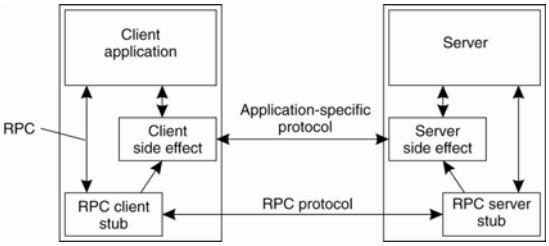
Subsistema RPC2

- Ejemplo un cliente inicia una conexión para solicitar la apertura de un archivo de video. Una vez que el cliente abre el archivo lo que se requiere entre cliente y servidor lo que se requiere es un canal donde se garantice que existirán ciertos retrasos (máximos y mínimos) de extremo a extremo al descargar el video (canal isócrono).

Subsistema RPC2

- El RPC2 permite que se establezca una conexión distinta para transferir a tiempo los datos de video al cliente.

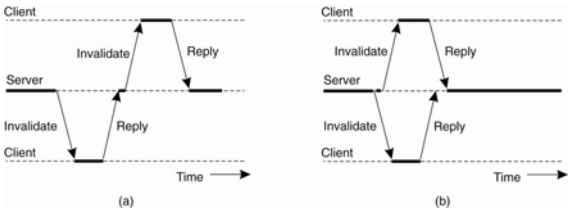
The RPC2 Subsystem (1)



Subsistema RPC2

- El RPC2 también da soporte a multitransmisión.
- En Coda, un tema importante de diseño es que los servidores no pierden de vista cuáles servidores tienen una copia local de un archivo.
- Cuando un archivo se modifica, un servidor invalida las copias locales (push) al notificar a los clientes apropiados mediante una RPC.
- Si un servidor puede notificar a un cliente a la vez, la invalidación puede llevarse algo de tiempo

The RPC2 Subsystem (2)



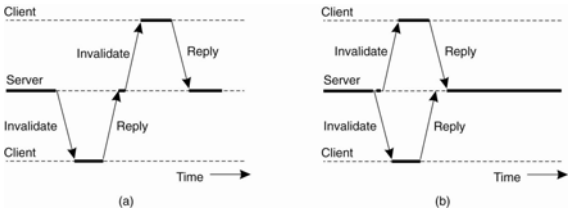
El subsistema RPC2

- Una solición mejor es: en lugar de invalidar las copias una por una, se envía un mensaje de invalidación a todos los clientes.
- Las RPC paralelas se implementan por medio del sistema MultiRPC (que es parte de RPC2). El receptor de una llamada MultiRPC no verá la fiferencia con una llamada RPC normal.

El subsistema RPC2

- Implementación:
  - Multiples RPC en paralelo: se invocan varias RPCs no bloqueantes, se aplaza el bloqueo hasta que todas las solicitudes han sido enviadas.
  - Establecer un grupo de multitransmisión y se envía una RPC a todos los miembros del grupo mediante multitransmisión IP.

The RPC2 Subsystem (2)



Asignación de Nombres

- Los nombres están organizados en un espacio de nombres jerárquico.
- Asignación de Nombres en NFS
  - La idea fundamental es proporcionar a los clientes un acceso completamente transparente a un sistema de archivos remoto que mantiene un servidor.

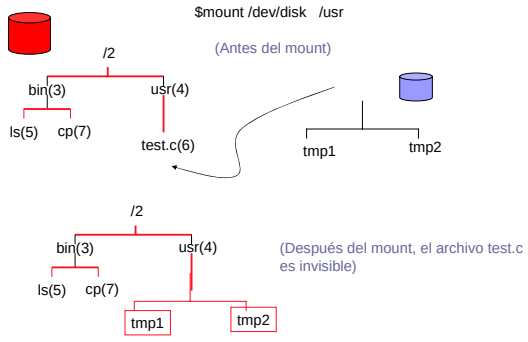
Asignación de Nombres

- Esto se logra permitiendo que el cliente sea capaz de montar un sistema de archivo remoto en su propio sistema de archivos local
- Un servidor exporta un directorio cuando coloca a éste y a sus entradas a disposición de sus clientes.

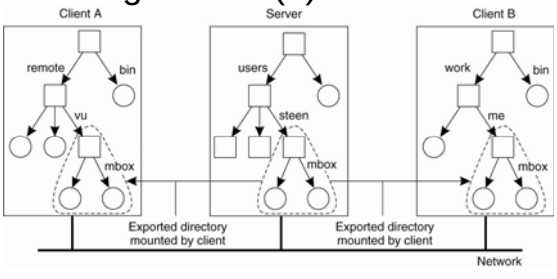
Mount

- Unix te permite crear o tener sistemas de archivos en otras particiones (que pudieran encontrarse en otros servidores) y “pegarlos”, a la jerarquía de directorios original utilizando un mecanismo denominado mounting (montura)

Mount



Naming in NFS (1)



/remote/vu/mbox

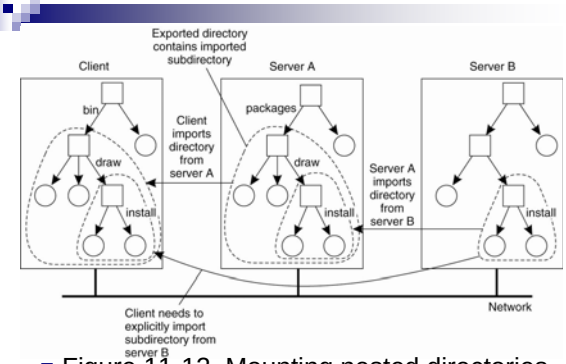
/work/me/mbox

Asignación de Nombres

- Un nombre de archivos depende de cómo organizan los clientes su propio espacio de nombres local y dónde monten los directorios exportados.
- Desventajas: compartir archivos puede ser difícil.
- Una solución: proporcionar a los clientes un espacio de nombres estandarizado. Por ejemplo, cada cliente puede usar el directorio /usr/bin para mostrar un conjunto de archivos disponibles para todos o el directorio /local.

Asignación de Nombres en NFS

- Un servidor de nombres puede exportar archivos por desde otros servidores, sin embargo no se permite que los exporte a los clientes. El cliente tendrá que exportarlos explícitamente del servidor que los contiene.
- Si un servidor puede exportar un directorio montado desde otro servidor tendría que regresar manejadores de archivos especiales que incluyan un identificador del servidor. Esto no está contemplado en los file handles de NFS.



- Figure 11-12. Mounting nested directories from multiple servers in NFS.

Asignación de Nombres

- Manejadores o descriptores de Archivo (file handle):
  - Es una referencia a un archivo localizado dentro de un sistema de archivos. El servidor que aloja el archivo crea este descriptor único. NFS4 (128 bytes)
  - Mientras el archivo exista debe tener el mismo manejador. El cliente puede almacenar el manejador que le devuelve el sistema de archivos remoto.

Asignación de Nombres

- Manejadores o descriptores de Archivo (file handle):
  - No son reutilizables por el servidor. Los clientes pueden usar el descriptor una vez que se ha borrado el archivo.
  - La mayoría de las operaciones sobre archivos se hacen con el manejador, se evita tener que utilizar el nombre.
  - El cliente puede acceder a un archivo independientemente de sus nombres actuales.

Automontaje

- Solución para los nombres de archivos diferentes: montar los sistemas de archivos remoto en el mismo o los mismos directorios para cada usuario.
- Problema: Decidir cuando deberá ser montado un sistema de archivos remoto (no lo debe hacer el usuario explicitamente).

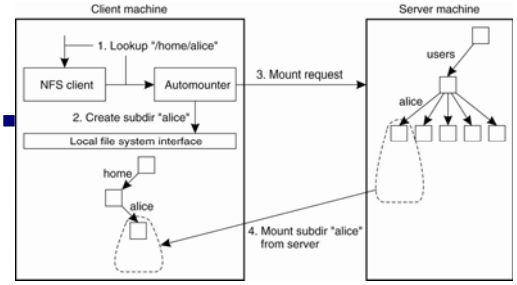
Automontaje

- Ejem: los directorios *home* de los usuarios están en un sistema de archivos remoto. */home/alicia.*
- *Sol:* montarlo cuando Alicia inicie una sesión en su estación de trabajo.
- Problema: si Alicia tiene acceso a /home/Bob. Este directorio debe montarse también cuando se inicie la sesión??

Automontaje

- Esto implicaría un enorme overhead al momento de iniciar las sesiones.
- Sol: El automontador que se encarga del montaje de acuerdo a la demanda.
- Se trata de un proceso distinto en la máquina cliente. La primera vez que se intente acceder a */home/alicia,* el kernel emite una operación de lookup al cliente NFS, quien a su vez remitirá la solicitud al automontador.

Automounting (1)



Sincronización (SAC)

- Semántica de Archivos Compartidos (SAC): Se refiere a cómo secuencializar las operaciones conflictivas sobre archivos de manera tal que no se lean valores obsoletos o se pierdan actualizaciones.

Sincronización (SAC)

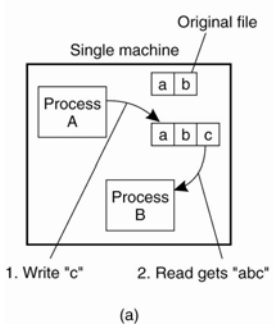
- Semántica de Unix:
  - Se usa en sistemas de un solo procesador que permiten a los procesos compartir archivos (UNIX)
  - Impone en todas las operaciones un orden absoluto en función del tiempo. Un “read” retorna el valor más reciente del dato.
  - Cada operación en un archivo es visible a todos los procesos en forma instantánea..

Sincronización (SAC)

- Semántica de Unix:
  - El sistema hace que se cumpla un ordenamiento en tiempo absoluto de todas las operaciones y siempre se regresa el valor más reciente.
  - En un sistema distribuido el desempeño de este método puede ser bastante pobre.

Semantics of File Sharing (1)

- Figure 11-16. (a) On a single processor, when a read follows a write, the value returned by the read is the value just written.



Sincronización (SAC)

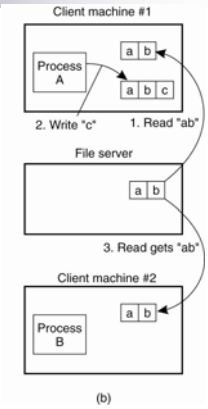
- En un sistema distribuido la semántica de UNIX es fácil de lograr, siempre que existe un solo servidor y los clientes no guarden datos en la memoria cache. Todas las operaciones conflictivas van al servidor el cual las procesa en secuencia.

Sincronización (SAC)

- Para mejorar el desempeño se puede permitir a los clientes tener copias locales de los archivos de uso frecuente en sus caches. Pero hay problemas de lecturas obsoletas.

Semantics of File Sharing (2)

- Figure 11-16. (b) In a distributed system with caching, obsolete values may be returned.



Sincronización (SAC)

- Para evitar el problema de lecturas obsoletas se puede:
  - Propagar inmediatamente las modificaciones al servidor: es un método conceptualmente simple pero ineficiente.
  - Relajar la semántica de compartimiento: **semántica de sesión**: los cambios a un archivo abierto son visibles sólo para el proceso o máquina que lo modificó. Los cambios serán visible al resto de los procesos o máquinas únicamente cuando se cierre el archivo.

Sincronización: Semántica de Archivos Compartidos

Semántica de sesión:

- Ningún cambio es visible a otros procesos hasta que el archivo se cierre.
- No todos los "reads" retornan el valor más reciente del dato.
- Si dos procesos tienen copias locales del mismo archivo y lo modifican al mismo tiempo, el resultado final depende de quién lo cierre primero.

Sincronización (SAC)

- La mayoría de los sistemas distribuidos implementan semántica de sesión. Esto significa que *aunque en teoría siguen el modelo de acceso remoto, la mayoría de las implementaciones usan caches locales, que efectivamente implementan el modelo de carga y descarga*.

Sincronización: Archivos Inmutables

- No hay forma de abrir un archivo para escritura. Las únicas operaciones permitidas son leer y crear archivos.
- Aunque es imposible modificar un archivo x, se puede reemplazar un archivo x con otro archivo. Después del reemplazo, x pasa a ser inaccesible, al menos con ese nombre.

Sincronización: Archivos Inmutables

- Se pueden actualizar los directorios ingresando nuevos archivos.
- Desaparece el problema de que un proceso esté leyendo el archivo y otro lo esté modificando (no se puede modificar un archivo).

Sincronización: Archivos Inmutables

- Es más fácil compartir archivos.
- ¿Qué sucede si dos procesos intentan reemplazar el mismo archivo a la vez? El resultado dependerá de la política empleada. El resultado final puede ser no determinista.
- ¿Qué sucede si un proceso reemplaza un archivo mientras otro lo está leyendo?
  - Una solucion es detectar que el archivo ha cambiado y que los intentos de lectura siguientes (del proceso que lo abrió) fallen.
  - Dejar que el lector continue leyendo el archivo viejo.

Sincronización: Transacciones

- Transacciones Atómicas:
  - Para acceder a un archivo o un grupo de archivos, el proceso ejecuta algún tipo de operación del estilo *BeginTransaction*, que indica al SD que las operaciones que siguen se tienen que ejecutar indivisiblemente.
  - Cuando termina el trabajo, el proceso hace *EndTransaction*.

Sincronización (SAC)

- Transacciones Atómicas:
  - La propiedad fundamental de este método es que el sistema garantiza que todas las llamadas contenidas dentro de la transacción serán realizadas en orden, sin ninguna interferencia de otras transacciones concurrentes.
  - Si dos o mas transacciones se inician al mismo tiempo, el sistema garantiza que el resultado final será el mismo como si se estuvieran ejecutando en algún orden secuencial no definido.

Semantics of File Sharing (3)

Method	Comment
UNIX semantics	Every operation on a file is instantly visible to all processes
Session semantics	No changes are visible to other processes until the file is closed
Immutable files	No updates are possible; simplifies sharing and replication
Transactions	All changes occur atomically

- Figure 11-17. Four ways of dealing with the shared files in a distributed system.

Sincronización: Bloqueos

- Los bloqueos permiten sincronizar el acceso a archivos compartidos.
- Existen diferentes tipos de bloqueos.
- Pueden existir diferencias en cuanto a la granularidad.

File Locking (1)

Operation	Description
Lock	Create a lock for a range of bytes
Lockt	Test whether a conflicting lock has been granted
Locku	Remove a lock from a range of bytes
Renew	Renew the lease on a specified lock

- Figure 11-18. NFSv4 operations related to file locking.

Sincronización (Bloqueos)

- En NFSv4 distingue los bloqueos de lectura y de escritura.
- La operación lock se utiliza para **solicitar** un bloqueo de lectura o escritura a lo largo de un intervalo consecutivo de bytes en un archivo. La operación es no bloqueante; si el cliente no puede obtener el lock el sistema le devuelve un mensaje de error y **el cliente tiene que sondear al servidor posteriormente**.
- Si el bloqueo está disponible, el cliente se bloquea.

Sincronización (Bloqueos)

- Alternativamente (con diferentes parámetros) el cliente puede solicitar al servidor que lo ponga en una lista de orden FIFO. Cuando el bloqueo se libera, el servidor otorga el siguiente bloqueo al próximo de la lista, siempre y cuando éste haya sondeado al servidor antes de la expiración de cierto tiempo.

Sincronización (Bloqueos)

- La operación lockt se utiliza para comprobar si existe un bloqueo conflictivo (antes de solicitarlo). Si existe un conflicto se le informa al cliente solicitante quien lo está provocando y sobre qué rango de bytes.
- locku se usa para eliminar un bloqueo a un archivo.

Sincronización (Bloqueos)

- Los bloqueos se otorgan por un tiempo específico determinado por el servidor. Existe una especie de contrato al otorgar un lock. Si el cliente no libera el contrato, el servidor le retira el bloqueo. Usando la operación renew, el cliente solicita al servidor renovar el contrato sobre el bloqueo.

Sincronización (Bloqueos)

- También existe una forma implícita de bloquear un archivo, conocida como **compartimiento de reservación**.
- Cuando un cliente abre un archivo, especifica el tipo de acceso que requiere (lectura, escritura o ambos) y qué tipo de acceso debe negar el servidor a otros clientes (Ninguno, lectura, escritura, ambos). Si el servidor no puede satisfacer los requerimientos del cliente la llamada *open()* fallará.

File Locking (2)

		Current file denial state			
		NONE	READ	WRITE	BOTH
Request access	READ	Succeed	Fail	Succeed	Fail
	WRITE	Succeed	Succeed	Fail	Fail
	BOTH	Succeed	Fail	Fail	Fail

(a)

- Figure 11-19. The result of an open operation with share reservations in NFS. (a) When the client requests shared access given the current denial state.

File Locking (3)

Current access state	Requested file denial state: NONE	READ	WRITE	BOTH
READ	Succeed	Fail	Succeed	Fail
WRITE	Succeed	Succeed	Fail	Fail
BOTH	Succeed	Fail	Fail	Fail

(b)

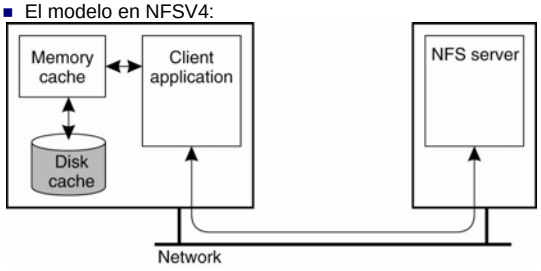
- Figure 11-19. The result of an open operation with share reservations in NFS. (b) When the client requests a denial state given the current file access state.

Consistencia y Replicación

- Almacenamiento en la memoria cache del cliente.
- Replicación del lado del servidor
- Replicación en sistemas de archivos punto a punto (peer-to-peer).

Almacenamiento en la memoria cache del cliente

- NFSV3 el almacenamiento del lado del cliente se dejó fuera del protocolo.
- El modelo en NFSV4:



Almacenamiento en la memoria cache del cliente

- En el cache se pueden guardar datos y metadatos de los archivos, manejadores de los archivos y directorios. Hay diferentes estrategias para mantener la consistencia de cada grupo.

Almacenamiento en la memoria cache del cliente

Con respecto al almacenamiento de datos:

- Cuando un cliente abre un archivo, se guarda en el cache el resultado de las operaciones *read().* Las operaciones *write()* pueden ser realizadas en el cache.
- Cuando el cliente cierra el archivo los datos almacenados en el cache deben devolverse al servidor (semántica de sesión)

Almacenamiento en la memoria cache del cliente

- NFS4
  - Una vez que parte de un archivo se encuentra almacenado en la memoria cache, sus datos pueden permanecer ahí, incluso después de cerrado el archivo, para que puedan ser usados por otros clientes
  - Si otro cliente abre un archivo -previamente cerrado- que haya sido guardado (en parte) en un cache local, el cliente debe revalidar de inmediato los datos del cache con el servidor.
  - En la invalidación se pregunta al servidor por la fecha de la última modificación del archivo, y si es necesario se invalida la copia local.

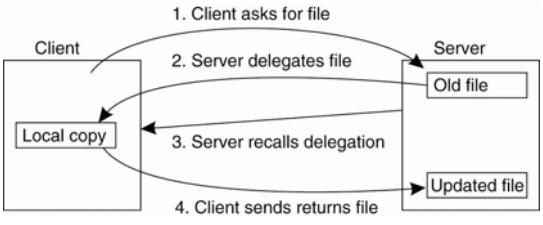
Almacenamiento en la memoria cache del cliente

- En NFS4 un servidor puede delegar sus derechos a un cliente cuando se abre un archivo.
- **Delegación Abierta**: ocurre cuando se permite que la máquina cliente maneje localmente las operaciones *open()* y *close()* de otros clientes en la misma máquina (se evitan contactos con el servidor)

Almacenamiento en la memoria cache del cliente

- Una consecuencia importante de delegar un archivo es que el servidor tiene que ser capaz de recordar la delegación (llamada de servidor al cliente, implementada por los mecanismos de RPC subyacentes)

Client-Side Caching (2)



Almacenamiento en la memoria cache del cliente

- NFS4
  - También se pueden guardar en el cache del cliente atributos. En este caso una política es que las modificaciones sean remitidas de inmediato al servidor.
  - También se pueden utilizar contratos para atributos y manejadores de archivos. Después de transcurrido cierto tiempo las entradas se invalidan y deben revalidarse antes de utilizarse otra vez.

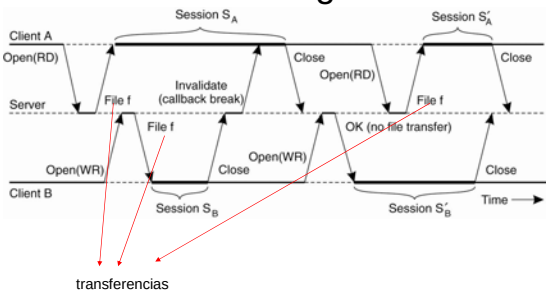
Almacenamiento en la memoria cache del cliente

- Coda: para lograr escalabilidad y tolerancia a fallos, los clientes siempre guardan en su cache los archivos completos. Cuando se abre un archivo para R/W se transfiere una copia completa del archivo a la memoria cache del cliente.
- La consistencia se mantiene mediante devoluciones de llamada por parte del servidor.

Almacenamiento en la memoria cache del cliente

- El servidor no pierde de vista cuáles son los clientes que poseen una copia del archivo y registra una **promesa de retorno de llamada**. Mientras esa llamada no se haga desde el servidor, la copia local sigue siendo válida, no obstante debe comprobarlo con el servidor.
- Cuando un cliente actualiza su copia del archivo, lo notifica al servidor quien a su vez envía un mensaje de invalidación al resto de los clientes (**ruptura del retorno de llamada**), ya no habrá más retorno de llamada.

Client-Side Caching in Coda



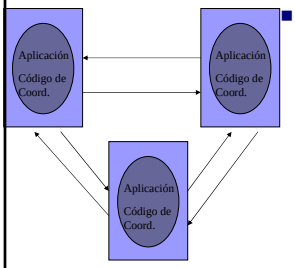
Replicación

- Razones para el servicio de réplicas:
  - Aumento de confiabilidad, al disponer de respaldos independientes de cada archivo.
  - Disponibilidad: permitir accesos aunque falle uno de los servidores.
  - Desempeño: repartir carga de trabajo entre varios servidores.

Replicación

- En Sistemas de Archivos es más común el uso de caches en el cliente porque la práctica muestra que el compartimiento de archivos es poco usual.
- Cuando tiene lugar es sólo para lectura de datos, por lo que el uso de memorias cache en el cliente es lo más adecuado.

Réplicas en sistemas Peer-to-Peer



- Todos los procesos desempeñan tareas semejantes, interactuando cooperativamente como iguales para realizar una actividad distribuida o cómputo sin distinción entre clientes y servidores: Gnutella, BitTorrent

Características

- Todos los nodos tienen conocimiento de los otros nodos.
- Cada nodo puede actuar como cliente y como servidor.
- Los “peers” son PCs usados para correo electrónico, navegar por el WEB, procesamiento de palabras, etc.
- Muchos de estos PCs se conectan a Internet desde casa. No tienen una dirección IP fija, estable. No están todo el tiempo conectados.
- Los usuarios no son normalmente entusiastas computistas. No saben de direcciones IP, protocolos, etc.

Redes Peer-to-Peer

- La tecnología P2P puede definirse como el uso coordinado de recursos geográficamente distribuidos en ausencia de un control central, basado en intercambios directos de información.
- Es un modelo de comunicación, en donde, clientes y servidores tienen las mismas capacidades y cualquiera puede iniciar la comunicación.

Filosofía de las Redes

- El **P2P** se basa principalmente en la filosofía e ideales de que todos los usuarios deben compartir. "El que más comparta, más privilegios tiene y más acceso dispone de manera más rápida a más contenido". Con este sistema se pretende asegurar la disponibilidad del contenido compartido, ya que de lo contrario no sería posible la subsistencia de la red.
- Aquellos usuarios que no comparten contenido en el sistema y con ello no siguen la filosofía propia de esta red, se les denominan "leechers"; los cuales muchas veces representan una amenaza para la disponibilidad de recursos en una red P2P debido a que únicamente consumen recursos sin reponer lo que consumen,

Procesos Peer to Peer

- Dichas redes son útiles para diversos propósitos. A menudo se usan para compartir archivos de cualquier tipo (por ejemplo, audio, video o software). Este tipo de red es también comúnmente usado en telefonía VoIP para hacer más eficiente la transmisión de datos en tiempo real.

Redes Peer to Peer

Algunos ejemplos de aplicación de las redes P2P:

- Intercambio y búsqueda de archivos. Quizás sea la aplicación más extendida de este tipo de redes. Algunos ejemplos son BitTorrent o la red eDonkey2000.
- Sistemas de telefonía por Internet, como Skype.
- A partir del año 2006 cada vez más compañías europeas y americanas, como Warner Bros o la BBC, empezaron a ver el P2P como una alternativa a la distribución convencional de películas y programas de televisión, ofreciendo parte de sus contenidos a través de tecnologías como la de BitTorrent.1
- Cálculos científicos que procesen enormes bases de datos.

Tomado de Wikipedia

Replicación en Sistemas de Archivos Peer to Peer

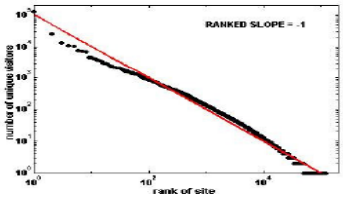
- La replicación desempeña un rol importante para acelerar las búsquedas y para equilibrar la carga de nodos.
- Una propiedad importante de estos sistemas es que todos los archivos son de lectura.
- Las actualizaciones consisten sólo en agregar archivos al sistema.

Replicación en Sistemas de Archivos Peer to Peer

- En sistemas no estructurados si se necesita un dato se busca en la red. Un nodo transmite la solicitud de búsqueda a sus vecinos, quienes a su vez retransmiten si es necesario.
- Una estrategia de réplica: en todos los nodos (imposible!! los nodos tienen capacidad limitada)

Replicación en Sistemas de Archivos Peer to Peer

- Enfoques:
  - Distribuir uniformemente ***n*** copias de cada archivo a través de toda la red. Ignora la popularidad de los archivos.
  - Replicar archivos de acuerdo a qué tan a menudo son buscados (que tan populares son). Pudiera resultar muy costoso localizar archivos no populares.
  - La replicación en sistemas peer-to –peer sucede naturalmente cuando los usuarios descargan archivos a partir de otros usuarios y luego los colocan a disposición de la comunidad.



El parámetro es la pendiente de la recta en escala logarítmica

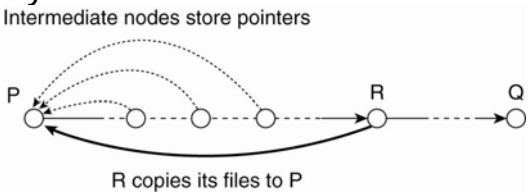
Replicación en Sistemas de Archivos Peer to Peer

- Sistemas estructurados: un método consiste en colocar réplicas en todos los nodos que están en la ruta desde el origen al destino.
  - Ventajas: Se descarga al nodo destino porque la mayoría de las réplicas se colocan en nodos cercanos a él, descargándolo en momentos donde exista una alta tasa de solicitudes.
  - Desventajas: No toma en cuenta la carga en los nodos, por lo que puede conducir a un sistema desbalanceado.

Replicación en Sistemas de Archivos Peer to Peer

- Sistemas estructurados: Otro método (Gopalakrishnan y colaboradores 2004) toma en cuenta la carga en los nodos a lo largo de la ruta de búsqueda.
- Se guardan réplicas de los archivos solicitados en el nodo origen de la búsqueda (si está más descargado que el nodo destino) y se guardan apuntadores en la memoria cache de los nodos que están en la ruta.

Structured Peer-to-Peer Systems



Preguntas del Capítulo

- Se requiere que un servidor de archivos que implemente nfsv3 sea sin estado?
- El uso de RPC2 es conveniente para flujos de datos continuos. Proporcione otro ejemplo en el cual tenga sentido utilizar un protocolo específico de la aplicación al lado de RPC?
- Suponga que el estado de denegación de un servicio en NFS es WRITE. ¿Es posible que otro cliente pueda abrir por primera vez con éxito el archivo y luego solicitar un bloqueo de escritura?

Preguntas

- Implementa NFS la consistencia de Entradas?
- Estipulamos que el NFS implementa el modelo de acceso remoto al manejo de archivos. Se puede argumentar que también implementa el modelo de carga y descarga? Explique por qué.
- Explique cómo resuelve Coda los conflictos de lectura y escritura en un archivo compartido entre múltiples lectores y un solo escritor.

Preguntas del Tema anterior

- Explique con sus propias palabras cuál es la razón principal para considerar modelos de consistencia débiles.
- Durante la explicación de los modelos de consistencia a menudo nos referimos al contrato entre el software y el almacen de datos ¿Por qué es necesario establecer dicho contrato?
- Qué tipo de consistencia utilizaría Ud. para implementar un mercado electrónico de acciones?
- Considere un buzón electrónico personal para un usuario móvil implementado como una BD distribuida de área amplia. Qué clase de consistencia sería la más adecuada.?

Preguntas del Tema anterior

- Para que la replicación activa funcione, en general, es necesario que que todas las operaciones se realicen en el mismo orden en cada réplica? ¿Siempre es necesario este ordenamiento?
- Un archivo es replicado en 10 servidores. Haga una lista de todas las combinaciones de quórum de lectura y de escritura que permite el algoritmo de votación.